Práctica 2
Heap y Arboles Generales

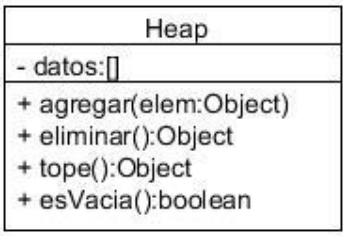
Ejercicio 1

Muestre las transformaciones que sufre una Max HEAP (inicialmente vacía) al realizar las siguientes operaciones:

a. Insertar 50, 52, 41, 54, 46
b. Eliminar tres elementos
c. Insertar 45, 48, 55, 43
d. Eliminar tres elementos.

Ejercicio 2

Considere la siguiente especificación de la clase **Heap**:



- El método **esVacia():bool** devuelve **true** si no hay elementos para extraer.
- El método **agregar(elem)** agrega un elemento en la heap y devuelve **true** si la inserción fue exitosa y **false** en caso contrario.
- El método **eliminar()** elimina el tope y lo retorna.
- El método **tope()** retorna el tope sin eliminarlo.

Además implemente un constructor en la clase **Heap** que tome dos parámetros, el primero un arreglo con el cual se crea e inicializa la Heap a partir de reordenar los elementos del mismo y el segundo un *boolean* que indique si se trata de una Max HEAP o Min HEAP (utilice **true** para indicar que es Max HEAP y **false** para Min HEAP).

Ejercicio 3

Se desea implementar una clase **Impresora** que ofrezca los siguientes métodos:

- **nuevoDocumento(documento: String)**: Encola un nuevo documento para imprimir.

- **imprime()**: Imprime por pantalla el documento almacenado más corto y lo elimina de memoria. Si no hay documentos en espera no hace nada.

Ejercicio 4

En base a la implementación de árboles generales proporcionada por la cátedra, complete la misma agregando los siguientes métodos:

a. **altura(): int** devuelve la altura del árbol, es decir, la longitud del camino más largo desde la raíz hasta una hoja.

   Pista: *El mensaje altura debe chequear si el árbol es una sola hoja o no. Si el árbol es una sola hoja, se devuelve 0. Si no, se utiliza el mensaje getHijos() para obtener la lista de hijos (recuerde que devuelve una lista de árboles hijos). Luego, debe iterar por cada uno de los hijos, guardando la máxima altura. A este valor se le debe sumar 1 y retornarlo.*

b. **nivel(Object dato):int** devuelve la profundidad o nivel del dato en el árbol. El nivel de un nodo es la longitud del único camino de la raíz al nodo.

   Pista: *Si el nodo raíz posee el mismo dato que pasado como parámetro, se retorna 0. En caso contrario, se debe buscar en cuales de los subárboles hijos se encuentra el dato (implemente el mensaje include (Object dato) en la clase Arbol General) y se debe retornar 1 más el nivel que arroje enviar el mensaje nivel() al subárbol que incluye el dato.*

c. **ancho():int** La amplitud (ancho) de un árbol se define como la cantidad de nodos que se encuentran en el nivel que posee la mayor cantidad de nodos.

   Pista: *Realice un recorrido por niveles. Encole inicialmente la raíz del árbol y luego una marca null (o el número de nivel) para indicar el fin de nivel. Mientras la cola no se vacía, itere. En cada iteración extraiga el tope de la cola, y con la operación getHijos() encole los mismos. Cuando encuentra la marca de fin de nivel cuente si los elementos del nivel es mayor a la máxima cantidad que poseía*.

Ejercicio 5

Sea una red de agua potable, la cual comienza en un caño maestro y el mismo se va dividiendo sucesivamente hasta llegar a cada una de las casas. Por el caño maestro ingresan 1000 litros y en la medida que el caño se divide, el caudal se divide en partes iguales en cada una de las divisiones. Es decir, si el caño maestro se divide en 4 partes, cada división tiene un caudal de 250 litros. Luego, si una de esas divisiones se vuelve a dividir en 3 partes, cada una tendrá un caudal de 83,3.

Usted debe implementar un método en la clase árbol general, que considerando que ingresan 1000 litros por el caño maestro, calcule el caudal de cada nodo y determine cuál es el mínimo caudal que recibe una casa.

Ejercicio 6

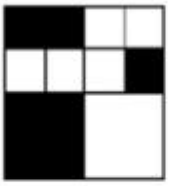
Un *quadtree* es una representación usada para cubrir un espacio cuadrado en dos dimensiones y posteriormente utilizado para determinar ciertas condiciones entre objetos en el mismo.

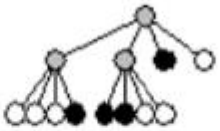
Un artista moderno trabaja con imágenes codificadas en *quadtrees*. El *quadtree* es un árbol 4-ario que codifica a una imagen con el siguiente criterio:

- Si toda la imagen tiene un mismo color, la misma es representada por un único nodo que almacene un dato que represente a ese color.

- En caso contrario, se divide la imagen en cuatro cuadrantes que se representan en el árbol como un nodo con 4 hijos, y cada hijo es la conversión de cada una de las partes de la imagen.

El artista desea saber cuántos pixeles de color negro posee una imagen dada. Usted debe implementar un método, que dado un *quadtree* y una cantidad total de pixeles, cuente cuantos pixeles de color negro contiene la imagen codificada en él.





Para el quadtree de la Figura, la salida del método sería 448

La figura muestra un ejemplo del árbol *quadTree* correspondiente a la imagen de la izquierda de 32 x 32 pixeles (1024 pixeles en total). Cada nodo en un *quadtree* es una hoja o tiene 4 hijos. Los nodos se recorren en sentido contrario a las agujas del reloj, desde la esquina superior derecha.